Cvičení 9

9.1 Úvod do Prologu, backtracking

Příklad 9.1.1 Spusťte a ukončete interpret jazyka Prolog.

Příklad 9.1.2 Načtěte rodokmen ze souboru `09_pedigree.pl` do prostředí Prologu. Soubor naleznete v ISu a v příloze sbírky. Formulujte vhodné dotazy a pomocí interpretru zjistěte odpovědi na následující otázky:

a) Je Pavla rodičem Filipa?
b) Je Pavla rodičem Vlasty?
c) Jaké děti má Pavla?
d) Má Adam dceru?
e) Kdo je otcem Petra?
f) Které dvojice otec-syn známe?

Příklad 9.1.3 Pro rodokmen ze souboru `09_pedigree.pl` naprogramujte následující predikáty:

a) `child/2`, který uspěje, jestliže první argument je dítětem druhého.
b) `grandmother/2`, který uspěje, jestliže první argument je babičkou druhého.
c) `brother/2`, který uspěje, jestliže první argument je bratrem druhého argumentu (mají tedy alespoň jednoho společného rodiče).

Příklad 9.1.4 Pro rodokmen ze souboru `09_pedigree.pl` naprogramujte predikát `stepBrother/2`, který uspěje, jestliže první argument je nevlastním bratrem druhého argumentu (mají tedy právě jednoho společného rodiče).

Příklad 9.1.5 Pro rodokmen ze souboru `09_pedigree.pl` napište predikát `descendant/2`, který uspěje, když je první argument potomkem druhého (ne nutně přímý). Bez použití interpretru určete, v jakém pořadí budou nalezeni potomci Pavly, když použijeme dotaz `?- descendant(X,pavla)`.

Jaký vliv má pořadí klauzulí a cílů v predikátu `descendant` na jeho funkci?

Příklad 9.1.6 Namodelujte osoby a rodičovské vztahy relacemi `daughter/2` a `son/2` v rodině Simpsonových.

Naprogramujte predikát `sister/2`, který uspěje, pokud osoba zadaná jako druhý parametr je sestrou osoby zadané jako první parametr (nikdo si není sám sobě sourozencem). Zformulujte dotaz, jehož úplným vyhodnocením se dozvíte všechny Bartovy sestry. Zkuste stejným predikátem zjistit, jaké sestry má Lisa.

9.2 Unifikace

Příklad 9.2.1 Které unifikace uspějí, které ne a proč? Jaký je výsledek provedených unifikací?

a) `man(X) = woman(X)`
b) `X = blue(Y)`
c) `green(X) = green(X,X)`
d) `X = man(X)`
e) `jmeno(X,X) = jmeno(Petr,novak)`
f) `weekend(day(saturday),day(sunday)) = weekend(X,Y)`
g) `weekend(day(saturday),X) = weekend(X,day(sunday))`

Příklad 9.2.2 Pro každou z následujících unifikací rozhodněte, jestli uspěje. Jestli ano, zapište její výsledek.

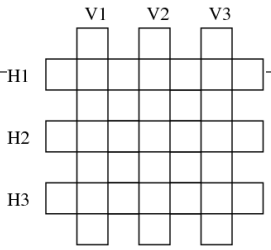
a) `monday = monday`
b) `'Monday' = monday`
c) `'monday' = monday`
d) `Monday = monday`
e) `monday = tuesday`
f) `day(monday) = monday`
g) `day(monday) = X`
h) `day(X) = day(monday)`
i) `day(monday,X) = day(Y,tuesday)`
j) `day(monday,X,wednesday) = day(Y,tuesday,X)`
k) `day(monday,X,wednesday) = day(Y,thursday)`
l) `day(D) = D`
m) `s(1,a(X,q(w))) = s(Y,a(2,Z))`
n) `s(1,a(X,q(X))) = s(W,a(Z,Z))`
o) `X = Y, P = R, s(1,a(P,q(R))) = s(Z,a(X,Y))`

Příklad 9.2.3 Unifikujte následující výrazy a zapište výslednou unifikaci.

a) `p(X)` a `p(f(Y))`
b) `p(X,f(X))` a `p(f(X),Y)`
c) `p(X,Y)` a `p(Z,Z)`
d) `p(X,8)` a `p(Y,Y)`
e) `p(f(X))` a `p(Z,Y)`
f) `p(f(Z),g(X))` a `p(Y,g(Y))`
g) `p(X,g(Z),X)` a `p(f(Y),Y,W)`

Příklad 9.2.4 Pro níže uvedenou databázi slov napište predikát `crossword/6`, který vypočítá, jak vyplnit uvedenou křížovku. První tři argumenty představují slova uváděná vertikálně shora dolů, druhé tři argumenty představují slova uváděná horizontálně zleva doprava.

```
word(astante,  a,s,t,a,n,t,e).
word(astoria,  a,s,t,o,r,i,a).
word(baratto,  b,a,r,a,t,t,o).
```




```
word(cobalto,  c,o,b,a,l,t,o).
word(pistola,  p,i,s,t,o,l,a).
word(statale,  s,t,a,t,a,l,e).
```

Příklad 9.2.5 Napište predikát `geq/2`, který jako argumenty dostane dvě čísla zadaná pomocí následníků a uspěje, pokud první číslo je větší nebo rovné druhému. Například dotaz `?- geq(s(0), s(s(0))).` neuspěje.

Příklad 9.2.6 Napište predikát `add/3`, který sčítá dvě přirozená čísla zadaná pomocí následníků. Například `?- add(s(0),s(s(0)),X).` uspěje s unifikací `X = s(s(s(0)))`.

Příklad 9.2.7 Napište predikát `mult/3`, který vypočítá součin dvou čísel zadaných pomocí následníků. V případě potřeby použijte predikát `add/3` definovaný v úloze 9.2.6. Příklad: dotaz `?- mult(s(s(0)),s(s(0)),X).` uspěje s unifikací `X = s(s(s(s(0))))`.

9.3 Backtracking a SLD stromy

Příklad 9.3.1 Uvažme následující program a dotaz `?- a.` Nakreslete odpovídající výpočetní SLD strom.

```
a :- b,c.
a :- d.
b.
b :- d.
c.
d :- e.
d.
```

Příklad 9.3.2 Uvažme následující program představující podmínky k dosažení bakalářského titulu (pravidla) a věci, které máte úspěšně za sebou (fakta). Nakreslete odpovídající výpočetní SLD strom pro dotaz `?- bcDegree.`

```
bcDegree :- courses, thesis.
courses :- programming, maths.
courses :- programming, exception.
exception :- deanAgrees.
exception :- rectorAgrees.

programming.
thesis.
deanAgrees.
```

Příklad 9.3.3 Uvažme následující databázi faktů:

```
r(a,b).
r(a,c).
r(b,d).
```

```
f1(a).
f1(X) :- f1(Y), r(Y,X).

f2(X) :- f2(Y), r(Y,X).
f2(a).

g1(a).
g1(X) :- r(Y,X), g1(Y).

g2(X) :- r(Y,X), g2(Y).
g2(a).
```

Zdůvodněte chování interpretru pro následující dotazy a nakreslete odpovídající SLD stromy.

1. `?- f1(X).`
2. `?- f2(X).`
3. `?- g1(X).`
4. `?- g2(X).`

Příklad 9.3.4 Uvažte následující program a dotaz `?- p(X,X).` Nakreslete odpovídající výpočetní SLD strom.

```
p(X,Y) :- q(X,Z), r(Z,Y).
p(X,X) :- s(X).
q(X,b).
q(b,a).
q(X,a) :- r(a,X).
r(b,a).
s(X) :- t(X,a).
s(X) :- t(X,b).
s(X) :- t(X,X).
t(a,b).
t(b,a).
```

Řešení

Řešení 9.1.1 *SWI-Prolog* spustíme jednoduchým voláním `swipl` z příkazové řádky. Základní predikáty pro práci s interpretrem jsou uvedeny níže.

- **`help/0`** Zobrazí základní nápovědu o použití nápovědy.
- **`help/1`** Zobrazí nápovědu k predikátu v argumentu.
- **`apropos/1`** Zobrazí predikáty, které mají v názvu nebo popisu zadaný výraz.
- **`consult/1`** Načte (zkompiluje) zdrojový kód ze zadaného souboru.
- **`make/0`** Znovu načte zdrojový kód ze změněných souborů.
- **`halt/0`** Ukončí interpret *SWI-Prolog*.
- **`trace/0`** Zapíná ladící mód a krokování výpočtu (existuje i predikát **`notrace/0`**, který vypíná krokování, ale nechává ladící mód).
- **`nodebug/0`** Vypíná krokování i ladící mód (existuje i predikát **`debug/0`**, který zapíná ladící mód, ale nezapíná krokování výpočtu).

Mějte na paměti, že příkazy v Prologu musí být ukončeny tečkou, jinak interpret očekává, že příkaz pokračuje.

Řešení 9.1.2

a) `?- parent(pavla,filip).`
`true.` Ano, je.

b) `?- parent(pavla,vlasta).`
`false.` Ne, není

c) `?- parent(pavla,X).`
`X = michal; X = filip; X = martin.` Pavla má děti Michala, Filipa a Martina.

d) `?- parent(adam,X), woman(X).`
`X = tereza.` Petr má jednu dceru, Terezu.

e) `?- father(X,petr).`
`X = michal.` Otcem Petra je Michal.

f) `?- father(X,Y), man(Y).`
`X = radek, Y = michal; X = radek, Y = filip; X = jirka, Y = martin;`
`X = michal, Y = petr; X = filip, Y = adam; X = adam, Y = vaclav.`

Řešení 9.1.3

a)
```
child(Child, Parent) :- parent(Parent, Child).
```

b)
```
grandmother(GM, GCH) :- woman(GM), parent(GM,CH), parent(CH,GCH).
```

c)
```
brother(Brother, Sibling) :-
  man(Brother),
  parent(Parent, Brother), parent(Parent, Sibling),
  Brother \= Sibling.
```

Řešení 9.1.4

```
stepBrother(Brother, Sibling) :- man(Brother),
  parent(Parent1, Brother), parent(Parent2, Brother),
  Parent1 \= Parent2, parent(Parent1, Sibling),
  Brother \= Sibling,
  parent(Parent3, Sibling), Parent3 \= Parent1, Parent3 \= Parent2.
```

Řešení 9.1.5

```
descendant(Desc,Anc) :- parent(Anc,Desc).
descendant(Desc,Anc) :- parent(Anc,Int), descendant(Desc,Int).
```

Dotaz na potomky Pavly vrátí osoby v následujícím pořadí: Michal, Filip, Martin, Petr, Adam, Tereza, Václav. Pořadí klauzulí v predikátu ovlivňuje způsob prohledávání stromu řešení. Kdybychom klauzule výše přehodili, našli bychom vzdálenější potomky (Petr, Tereza) dříve než jejich korespondující rodiče (Michal, Adam). Pokud bychom přehodili cíle ve druhé klauzuli, způsobíme zacyklení výpočtu, neboť interpret se bude snažit nalézt nejdříve ty vzdálenější a vzdálenější potomky.

Řešení 9.1.6

```
daughter(marge, lisa).
daughter(homer, lisa).
daughter(marge, maggie).
daughter(homer, maggie).
son(marge, bart).
son(homer, bart).
```

Předpokládejme, že sestra je taková dcera, která se svým sourozencem sdílí alespoň jednoho rodiče. Implementujme predikát `sister/2` následovně:

```
sister(Person, Sister) :- daughter(Parent, Sister),
    daughter(Parent, Person), Sister \= Person.
sister(Person, Sister) :- daughter(Parent, Sister), son(Parent, Person).
```

Výše uvedené řešení vrátí na dotaz `?- sister(bart, X).` všechny Bartovy sestry, avšak každou dvakrát (každá je totiž sestrou i přes matku i přes otce). Vypisovat každou sestru pouze jednou by bylo poněkud složitější, bylo by potřeba podrobně rozepsat všechny možné případy.

Sestry Lisy zjistíme dotazem `?- sister(lisa, X).` Poznamenejme ještě, že vypsání Lisy jako své vlastní sestry zabraňuje podmínka neunifikovatelnosti na konci první klauzule řešení.

Řešení 9.2.1

a) Neuspěje, protože `man/1` a `woman/1` jsou různé predikáty (mají různý název).
b) Uspěje, výsledná unifikace `X = blue(Y)`.
c) Neuspěje, protože `green/1` a `green/2` jsou různé predikáty (mají různou aritu).
d) Uspěje s unifikací `X = man(X)`. Unifikace je však nekorektní, protože vznikne nekonečný term. Prolog však kontrolu výskytu nedělá (je to náročnější problém, jak se zdá).

e) Uspěje, výsledná unifikace `X = Petr, Petr = novak`.
f) Uspěje, výsledná unifikace `X = day(saturday), Y = day(sunday)`.
g) Neuspěje, unifikace selhala na podmínce `saturday = sunday`.

Řešení 9.2.2

a) Uspěje, výsledná unifikace je prázdná.
b) Neuspěje, termy mají různá jména.
c) Uspěje, výsledná unifikace je prázdná.
d) Uspěje, výsledná unifikace `Monday = monday`.
e) Neuspěje, termy mají různá jména.
f) Neuspěje, termy mají různá jména a arity.
g) Uspěje, výsledná unifikace `X = day(monday)`.
h) Uspěje, výsledná unifikace `X = monday`.
i) Uspěje, výsledná unifikace `X = tuesday, Y = monday`.
j) Neuspěje, unifikace selhala na podmínce `tuesday = wednesday`.
k) Neuspěje, termy mají různé arity.
l) Uspěje s výslednou unifikací `day(D) = D`. Tato unifikace je však nekorektní, protože vznikne nekonečný term (Prolog kontrolu výskytu nedělá).
m) Uspěje, výsledná unifikace `X = 2, Y = 1, Z = q(w)`.
n) Uspěje s unifikací `X = Z, Z = q(Z), W = 1`. Unifikace je však nekorektní, protože vznikne nekonečný term.
o) Uspěje s unifikací `X = Y, Y = P, P = R, R = q(R), Z = 1`. Unifikace je však nekorektní, protože vznikne nekonečný term.

Řešení 9.2.3

a) `X = f(Y).`
b) Výrazy nejsou unifikovatelné (unifikace v interpretru sice uspěje, ale není korektní).
c) `X = Y, Y = Z.`
d) `X = Y, Y = 8.`
e) Výrazy nejsou unifikovatelné, predikáty na nejvyšší úrovni mají různou aritu.
f) `X = Y, Y = f(Z).`
g) `X = W, W = f(g(Z)), Y = g(Z).`

Řešení 9.2.4 První uvedené řešení je sice jednoduché a relativně intuitivní, je však značně neefektivní.

```
crossword( V1, V2, V3, H1, H2, H3 ) :-
    word( V1, _V11, V12, _V13, V14, _V15, V16, _V17 ),
    word( V2, _V21, V22, _V23, V24, _V25, V26, _V27 ),
    word( V3, _V31, V32, _V33, V34, _V35, V36, _V37 ),
    word( H1, _H11, H12, _H13, H14, _H15, H16, _H17 ),
    word( H2, _H21, H22, _H23, H24, _H25, H26, _H27 ),
    word( H3, _H31, H32, _H33, H34, _H35, H36, _H37 ),
    V12 = H12, V14 = H22, V16 = H32,
    V22 = H14, V24 = H24, V26 = H34,
    V32 = H16, V34 = H26, V36 = H36.
```

Řešení se dá napsat i trochu lépe, když namísto explicitních unifikací použijeme stejné proměnné už v predikátech pro slova:

```
crossword2( V1, V2, V3, H1, H2, H3 ) :-
    word( V1, _V11, V12, _V13, V14, _V15, V16, _V17 ),
    word( V2, _V21, V22, _V23, V24, _V25, V26, _V27 ),
    word( V3, _V31, V32, _V33, V34, _V35, V36, _V37 ),
    word( H1, _H11, V12, _H13, V22, _H15, V32, _H17 ),
    word( H2, _H21, V14, _H23, V24, _H25, V34, _H27 ),
    word( H3, _H31, V16, _H33, V26, _H35, V36, _H37 ).
```

Řešení 9.2.5

```
geq(_,0).
geq(s(X),s(Y)) :- geq(X,Y).
```

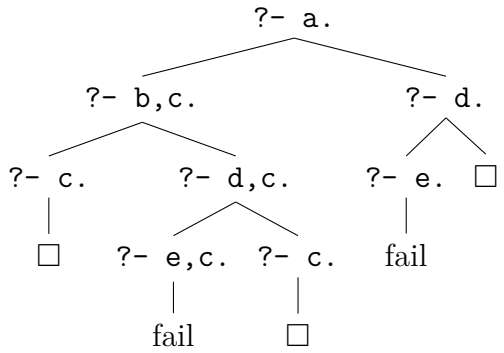
Řešení 9.2.6

```
add(0,X,X).
add(s(X),Y,s(Z)) :- add(X,Y,Z).
```

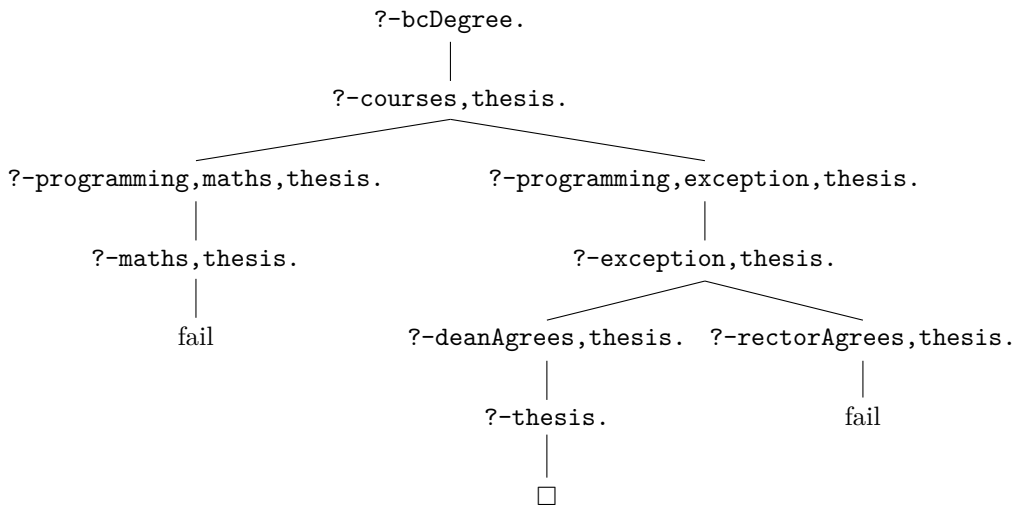
Řešení 9.2.7

```
mult(_X,0,0).
mult(X,s(Y),Z) :- mult(X,Y,Z1), add(X,Z1,Z).
```

Řešení 9.3.1

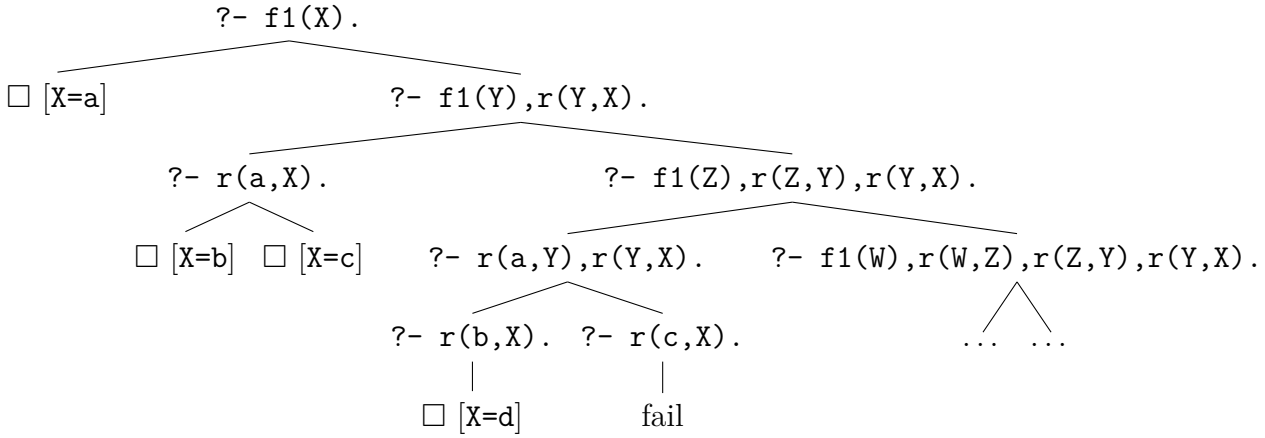


Řešení 9.3.2



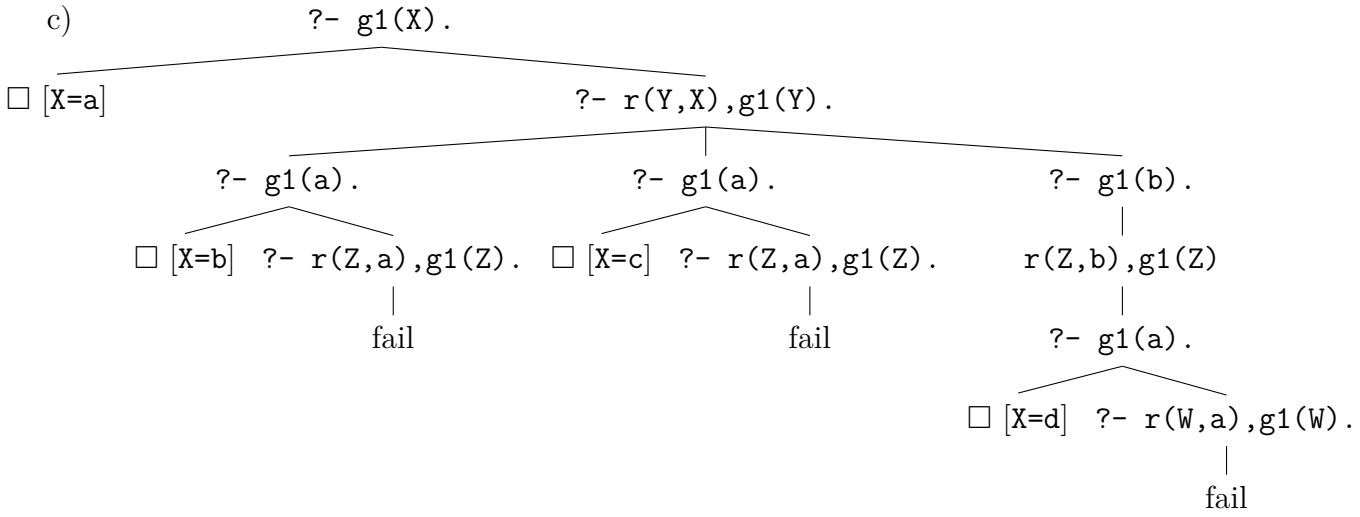
Řešení 9.3.3

a)



b) Výsledný SLD strom bude stejný jako v případě a), jenom každý podstrom, který má v kořeni výraz začínající predikátem `f1/1` bude zrcadlově převrácený. To však znamená, že nejlevější větev bude nekonečná, a tedy interpret bude cyklit.

c)



d) Výsledný SLD strom bude stejný jako v případě c), jenom každý podstrom, který má v kořeni výraz začínající predikátem `g1/1` bude zrcadlově převrácený. To znamená, že

uvedená řešení budou nalezena v pořadí `X=b`, `X=c`, `X=d`, `X=a`.

Řešení 9.3.4